NELC / TR 2001

NELC / TR 2001

12

ADA030902

A CHARGE-COUPLED DEVICE (CCD) MEMORY FOR NAVY C^3 SYSTEMS

30 July 1976

Research and Development, July 1975 to July 1976

Prepared for
NAVAL SEA SYSTEMS COMMAND
Washington, DC 20362

NAVAL ELECTRONICS LABORATORY CENTER
SAN DIEGO, CALIFORNIA 92152

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NELC Technical Report 2001 (TR 2001)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) A CHARGE-COUPLED DEVICE (CCD) MEMORY FOR NAVY C^3 SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED (9) Research and Development Jul[redacted]75 – Jul[redacted]76
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) J. J. Symanski (14) NELC/TR-2001		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Electronics Laboratory Center San Diego, CA 92152		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62721N; F21241; SF21241401 (NELC N712)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command Washington, DC 20362		12. REPORT DATE (11) 30 Jul[redacted]76
		13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 40p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited (16) NELC-N712, SF21-241

(17) SF21-241-401

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer memories
Charge-coupled memories

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A prototype memory system using 16 thousand-bit charge-coupled devices and a microprocessor controller has been designed and tested. Features include a modular, expandable architecture, error detection and correction, NTDS input and output, and SHP packaging.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

403 940

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

OBJECTIVE

Navy electronic systems are becoming more complex, more costly, and more difficult to maintain. Command control and communication systems require large amounts of data and program storage. Advances are being made in solid-state memory technologies which will benefit the Navy systems. The objective of this effort was to investigate the use of charge-coupled devices for computer memory systems and also to develop a controller which will be useful for charge-coupled device and other memory technologies.

RESULTS

A modular, media-independent memory system was designed and prototype hardware was fabricated and tested. A CCD module, for use in the memory system, was designed and a partially populated module was tested. Over 1200 hours of continuous operation indicated a CCD error rate of about 1×10^{-11} per bit. The system error rate was even lower.

RECOMMENDATIONS

1. Continue to test and to incorporate new solid-state devices such as CCDs, magnetic-bubble devices (MBDs), and metal nitride-oxide semiconductors (MNOS) into memory systems.

2. Continue the development of data-base management and information-retrieval techniques for memory systems.

3. Utilize advanced solid-state memory equipment to enhance the effectiveness of the Navy C^3 systems.

ADMINISTRATIVE INFORMATION

This work was performed for the Naval Sea Systems Command (03416) under Program Element 62721N, Project F21241, and Task Area SF21241401 (NELC N712) by members of the Communications Processing Division. This report covers work from July 1975 to July 1976 and was approved for publication on 30 July 1976.

The author expresses his appreciation to the following Code 3200 engineers for their many helpful comments and suggestions: Tom Keenan, Project Manager; Bob Ebert, Gary Huckell, and Mike Hallman on MIMC and CCD module system design; Russ Keefer for hardware logic design and fabrication details; and Henry Gok for the programming of the MIMC monitor and the CCD test program.

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1. INTRODUCTION

Disks and drums are presently used for secondary (mass) storage in computer systems. Their mechanical nature leads to problems in access time, maintenance, and reliability. In a commercial environment, the cost per bit is still competitive with semiconductor random-access memories (RAM) but the more harsh military environment brings the cost up to approximately 0.25 cents per bit for about 15 million bits of drum storage (6.25-ms access). The cost for an individual system will vary depending upon the support circuitry such as the controller, power supplies, and cabinet.

Recent advances in semiconductor technology have brought the charge-coupled device (CCD) to the point of competition with moderate capacity (a few million bits) drums and fixed-head disks (FHD). This report covers the development of a modular memory subsystem suitable for use in the Navy C^3 systems having requirements for secondary storage. Included in this subsystem is a microprocessor controller which is capable of being programmed in order to modify its operational characteristics to suit a particular task.

Since there are under development several other technologies which will be used for mass storage in the next five to ten years, the memory subsystem to be described has been called the Media-Independent Memory Controller (MIMC). The hardware has been designed and fabricated with the goal of using it to test prototype devices and architectural concepts for charge-coupled devices, magnetic-bubble devices, metal nitride-oxide semiconductors, and any other promising technologies, as time and funding will allow. Section 2 will describe the MIMC hardware, architecture, operation, and other characteristics.

Section 3 will describe the software-development aids, or monitor, which have been written to allow a programmer to communicate with the processor and to control peripherals. An NELC Technical Note covers the operations available in the MIMC monitor and will serve as an instruction manual for a programmer wishing to use the system.

Section 4 describes the philosophy, architecture, and operating characteristics of the CCD module designed for use in this system. The primary goal in the development of this module was to achieve very high reliability of data with maximum modularity and self-maintenance capabilities.

The results obtained from the assembly and test of a partially populated module are discussed in Section 5. The software and the various types of tests are described. Several weeks of constant reading and writing of the 256 thousand bits resulted in less than ten errors. All of these were corrected by the EDAC circuitry so that the system actually saw no errors. Calculations show that there is an error rate of about 1.2×10^{-11} for the CCD system.

Section 6 of this report summarizes the work and makes recommendations for improvements and future work.

SECTION 2. THE MEDIA-INDEPENDENT MEMORY CONTROLLER (MIMC)

BACKGROUND

Several technologies are under development which promise a solid-state, low-cost alternative to electromechanical data-storage equipment. Three of the most promising are charge-coupled devices (CCDs), magnetic-bubble devices (MBDs), and electron-beam devices (EBDs). Each of these devices has advantages and disadvantages. These will not be discussed in this report since the literature already contains many articles on this subject.

Another development which is rapidly gaining momentum is the microprocessor. This development also has many facets and there are dozens of microprocessors available.

Of particular interest, in this investigation, is the bipolar microprocessor slice because of its speed and adaptability to Navy systems. The use of microprocessors enables the designer to implant significant intelligence in the equipment he designs and still retain flexibility and expansion capability without major changes in hardware.

Concurrently, advances are being made in the area of software techniques which will improve the performance and capabilities of data-base systems. These new software approaches, combined with more intelligent hardware and high-speed, non-mechanical storage devices, will appreciably enhance reliability and performance of military systems.

The Media Independent Memory Controller (MIMC) is an attempt to combine these diverse developments into a single unit to be used to develop and test concepts and hardware. MIMC is aimed at providing multiple-media (electronic or magnetic) handling capability, advanced data-base management techniques, and a useful degree of localized intelligence for improved normal operating performance, as well as a significant self-maintenance and diagnostic capability for fault location and repair.

To aid the cost-effective application of these new developments, this equipment adopted the use of standard hardware from the beginning. The Super 2A card, developed as part of the Navy Standard Hardware Program (SHP), has been used in the prototype hardware. Several cards which are in use in another development, the Submarine Satellite Information Exchange System (SSIXS), are used in MIMC to simplify logistics and training problems. The various cards will be described in later sections of this report.

MIMC ARCHITECTURE

MIMC has a bus-structured architecture. Figure 1 shows a simple block diagram of the system. The bus definition is basically the same as the bus used in SSIXS 8080-based hardware. The control, power, address, and data lines are the same with 32 bidirectional data lines rather than eight. Details of the operation of the bus can be obtained from a description of the SSIXS hardware. The basic input-output (I/O), random-access memory (RAM), and read-only memory (ROM) boards are the same cards used in the SSIXS

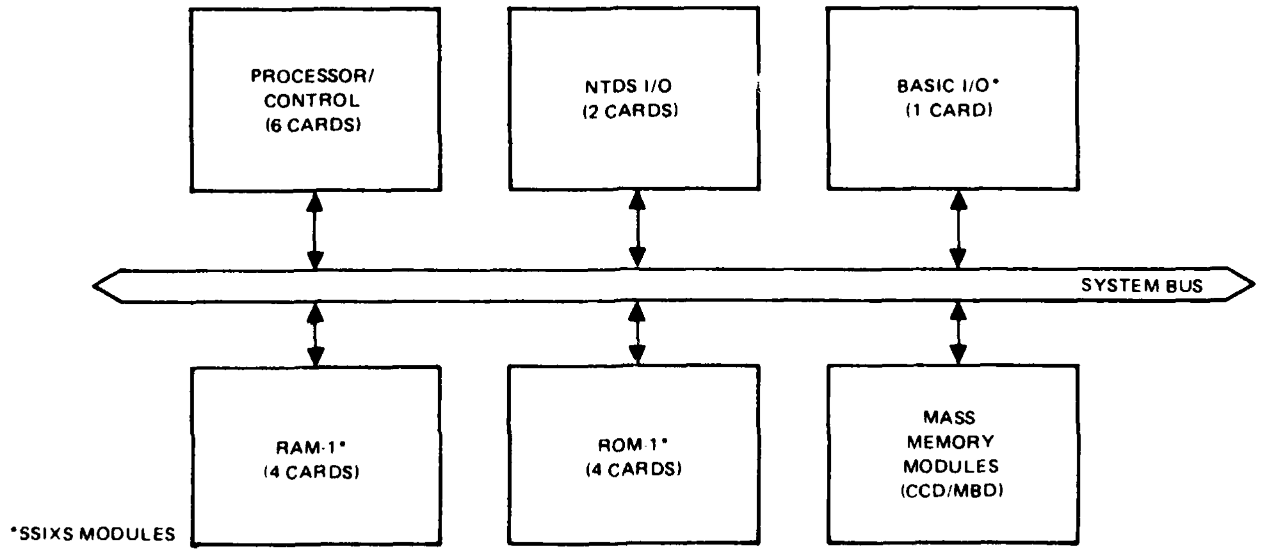


Figure 1. MIMC architecture.

hardware. The processor/control module consists of six cards which are unique to this system. Two of these six are identical (RALU cards, upper and lower). These two cards each contain four, four-bit slice microprocessors; the Monolithic Memories 57 6701, plus look-ahead carry generators, and heavy-duty bus drivers and receivers. Three cards, control 1, 2, and 3, contain the general system-timing circuits, the bus-control circuitry, address latches, and microcode ROMs, which interpret instructions and generate control signals for the microprocessor cards. Interrupt circuitry is also contained on these cards. See table 1 for a list of the modules used in a minimal system.

The debug card is used to display data on the 32-bit bus in hexadecimal format. There are also switches which allow manual entering of data or instructions for trouble-shooting and initializing the system.

The operation of the processor/control section will be explained. Basically, it is a 32-bit, microprogrammed processor with 16 working registers, a 'Q' register, and an arithmetic logic unit (the 6701, four-bit slice), a separate status register, memory-address register, vectored interrupts, bit-testing capability, and the timing circuitry to control the bidirectional data bus.

TABLE 1. MIMC MINIMUM SYSTEM.

Number of Cards	Card Type	Description
2	RALU – Microprocessor	4-6701, four-bit slices
1	Control 1	Instruction decode
1	Control 2	Address latches, drivers, status register, bit testing
1	Control 3	Clock, timing, bus control, interrupt circuitry
1	Debug	Panel control, hexadecimal display and drivers, bus interface
1	NTDS input	16-bit NTDS, line timing-bus, interface, etc
1	NTDS output	16-bit NTDS, line timing-bus, interface, etc
1	Basic I/O	SSIXS module
4	RAM	SSIXS module, 2k or 8k bytes each card
4	ROM	SSIXS module, 2k or 8k bytes each card

OPERATION

The heart of the MIMC is contained in the processor/control section. A block diagram of this section is shown in figure 2. A more detailed diagram of the 6701 four-bit slice is shown in figure 3. A brief description of the operation will be given below. For more details, the reader is referred to the logic schematics, microcoding lists and timing charts.

Instructions to be executed are stored in either RAM or ROM modules. An instruction is fetched by putting the contents of the program counter (register F) on the bus and loading the memory-address register. The first state of each instruction (state zero) reads the instruction off the data bus, loads the operation code (bits 24 to 31) into the microcode

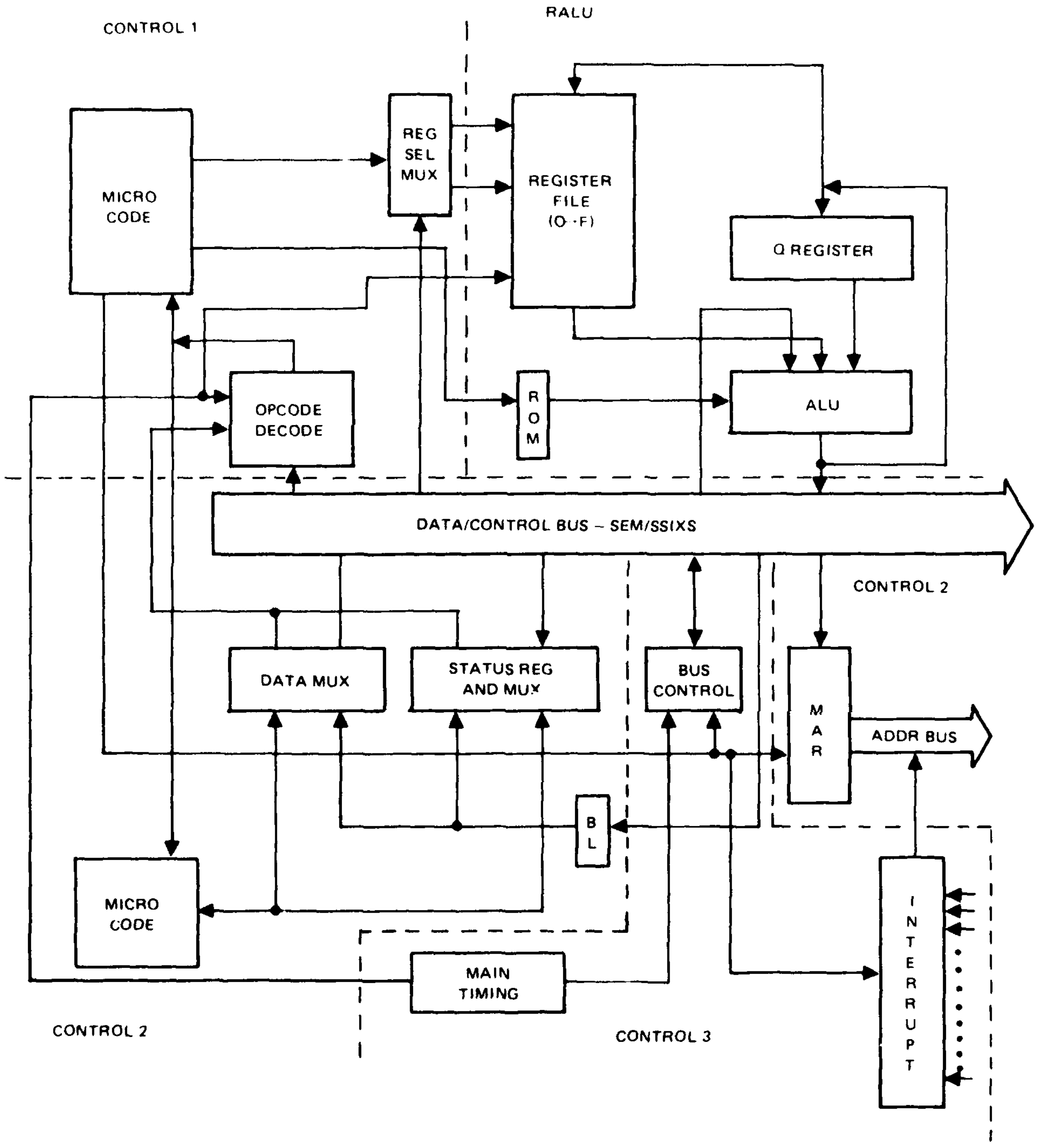


Figure 2. MIMC processor/control block diagram.

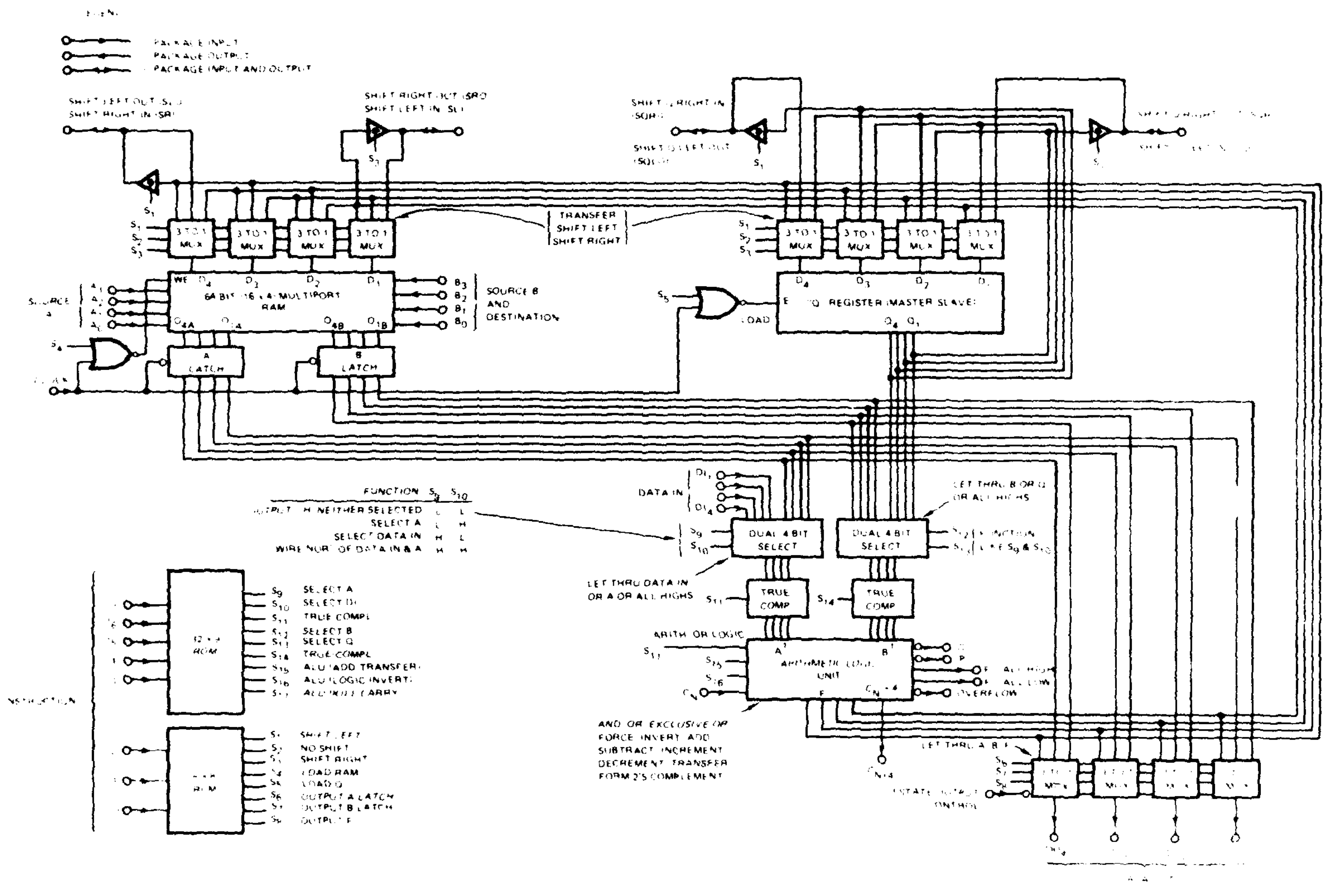


Figure 3. Expandable 4-bit slice, 40-pin package.

address circuitry, loads bits 16 to 23 into the register-selection multiplexer, and bits zero to 15 into the lower half of the operand register (register I). The upper half of register I is forced to zero during state zero. (See Appendix A for a description of the instruction format.)

The microcode address circuitry translates the operation code into a starting address for the microcode ROM. Depending upon the operation code, the microcode address circuitry will step through several microinstruction addresses until the end of that instruction is reached. One microcycle requires 250 to 500 nanoseconds.

For instance, the first state of microinstruction may put the program counter on the data bus and load the memory address register to start the next instruction-fetch cycle. The second microcycle of that instruction might load one register with the contents of another and end the operation. The next microcycle would be a state zero which would begin the next instruction execution.

In the instruction just discussed, the registers involved would be selected by the register-select multiplexer shown in figure 2.

There are instructions which test if a certain bit of the status register or a working register is set. The bit Ø→15 can be selected by the B field of the instruction (bits 20 to 23). The data multiplexer selects the specified bit from the bus while the status register/multiplexer circuitry allows the status register to be checked. The Bit Latch (BL) stores the four-bit hexadecimal code to select the bit position of interest.

Other elements of the control circuitry are the main timing, which includes a crystal oscillator, and the interrupt circuitry, which facilitates the vectored interrupt operation.

INPUT/OUTPUT

Input/output operations are performed exactly as they are in the SSIXS 8080 system. I/O ports are addressed on the normal address lines and the control lines indicate an I/O operation. The SSIXS I/O-1 card is used to communicate over RS-232 or 188-C interfaces. There are other cards designed for the SSIXS system which can also be used in MIMC such as, I/O-2 which is an AN/UYK-20 serial I/O, and I/O-3 which is a REMEX paper-tape reader and punch.

Two NTDS cards have been designed for MIMC to interface the bus with a 16-bit, NTDS, slow or fast channel. One card is required for input and one for output. The handshaking required for the NTDS interface is done on the card. The card can be set up to act as a peripheral or as a computer on both input and output. The cards can also be connected in parallel to yield a 32-bit NTDS interface. The cards have the "subchannel" addressing scheme; that is, each channel fills 16 addresses (for instance, addresses 40 to 4F). Presently, only the lower eight addresses are used; the upper eight are for expanded functions and are not yet assigned. Table 2 shows the functions selected with the eight addresses.

TABLE 2. MIMC NTDS FUNCTIONS.

Output Function	Code
Output Data	Ø
Output Command	1
Send Status Word	2
Clear Interrupt Request	3

TABLE 2. (Continued).

Output Function	Code
Not Used	4
Not Used	5
Enable Buffer Ready Interrupt	6
Disable Buffer Ready Interrupt	7

Input Function	Code
Input NTDS Data	Ø
Input NTDS Command	1
Send Status	2
Clear Interrupt Request	3
Enable Data Interrupt	4
Disable Data Interrupt	5
Enable Command Interrupt	6
Disable Command Interrupt	7

INTERRUPTS

MIMC presently has 16 interrupt channels. They are prioritized with channel 0 having the highest priority. Interrupts are vectored. That is, each interrupt channel forces a jump to a certain memory address. At that address are instructions which respond to the interrupt received. The appropriate response is determined by the programmer.

All interrupts can be enabled or disabled by software instructions. Individual channels can only be enabled or disabled by selecting each channel under software control.

An interrupt can occur at the start of each new instruction if interrupts are not disabled. With the use of the appropriate instructions, the program counter will automatically be stored and reloaded for normal operation.

SECTION 3. PROGRAMMING THE MIMC

THE INSTRUCTION SET

In this prototype hardware, only a basic set of 67 instructions has been implemented. The adopted format and definitions are described in Appendix A. A list of the instructions along with a brief description is given in Appendix B. The format has been kept simple and relatively straightforward. The basic classes of instructions (loads, stores, logical, arithmetic, shifting, jumps, and compares) have been implemented. Special instructions concerned with interrupts, stack operations, I/O, and other functions have also been included. With the 8-bit operation code, we have the possibility of 256 instructions. Since the "right" instructions depend upon the application (and opinion), we have left plenty of room for increasing the instruction set.

THE MONITOR

A minimum MIMC system has been assembled and interfaced with a LSI 7700 display terminal. In order to develop test programs and to exercise the hardware, a monitor program was developed and written into PROM so that it is always resident. With this monitor program, other programs can be developed and communication with other hardware is possible.

The present system has only 2048 words of PROM (address 0000 through $07FF_{16}$) and 2048 words of RAM (addresses 0800 thru 0FFF). In a larger system, the storage areas, stack-starting addresses, and the like, could be changed to utilize the larger memory.

The monitor was written with several objectives in mind: to include all possible routines that are feasible to implement and which are useful to the user; to enable the user to make use of the subroutines used by the monitor; and to have software breakpoint capabilities.

Some of the routines in the monitor were taken and modified from the 8080 and 642-B monitors. The present routines can handle two peripherals in addition to the console cathode-ray tube terminal: the REMEX paper-tape punch/reader and a line printer. A routine to handle the PRO-LOG PROM programmer has been developed and can be loaded and executed from RAM. Future routines to be implemented when time permits include a card-reader and handler, and a disk handler.

Most subroutines used by the monitor are written in such a way as to allow users to make good use of them. In other words, most of the monitor subroutines preserve all registers except the status, accumulator "Q", and the registers used during input. Also, most of the subroutines are I/O related. These subroutines allow the user to program the I/O structure by eliminating the details one encounters when programming input-output.

A software breakpoint, although not as cleanly implementable as a hardware breakpoint, is a very powerful tool for debugging and checking out software. This is a mechanism to enable the user to "break" from a certain location in his program and to allow him access to the current registers at that "breakpoint" of the program. The user is allowed to change the content of the registers (if he wishes) and to continue his program right where he left it (at the breakpoint location). This gives the user the opportunity to debug or checkout his program by stepping through it and observing the registers.

The monitor makes good use of the cathode-ray tube terminal. A backspace key ("∧") is allowed in some routines to allow the user to retype the last character. In EDIT MODE, a user is allowed to edit the data on the terminal screen using the keyboard and cursor control in a nonconversion mode with the MIMC. When the user has completed editing on the terminal screen, he can then send everything on the screen in one block to the memory in the MIMC.

Additional functions or routines may be temporarily added to the MIMC monitor without having to be stored in PROM. These routines can be loaded into RAM and a flag can be set at the location in RAM to designate the point where the comparison test of the input character versus function character is to be continued. A background program can be running in the MIMC at the same time a user is exercising MIMC. For example, a program to test CCD memories may be running in the MIMC at the same time a user is debugging his program.

During the period when MIMC is waiting for input from the keyboard, it executes one pass of the test of the CCD memory. After this pass, it polls the keyboard for any inputs and, if there are none, executes another pass of the test. If an input is received from the keyboard it will not be processed.

Another way to implement this background routine (which is not the way it is implemented in the current monitor) is to have the monitor on interrupts. That is, anytime the user types on the keyboard, an interrupt is generated which instructs the MIMC to process the keyboard input. The background program in this case could run continuously without its having to constantly poll the keyboard for input.

The monitor utilizes two stacks, one is for the monitor and the other is for the user programs. The monitor stack is always reset to its beginning each time the MIMC goes to RDY. The program stack is reset to its beginning each time the user types "INT" or "CLEAR SCREEN" or "EXECUTES" a program.

DESCRIPTION OF MONITOR FUNCTIONS

In the following paragraphs, brief descriptions of the functions available in the monitor are given. The format of the typed-in instructions is shown in capital letters, underlined. When the space bar is actuated, the operation is executed.

1. DUMP   dumps memory contents starting from a specified address XXXX entered by the user.

D XXXX	dumps 1/4 screen (6 lines)	24 locations
D XXXXH	clear screen & dump 1/2 screen (12 lines)	48 locations
D XXXXF	clear screen & dump full screen (24 lines)	96 locations

2. ENTER   allows the user to enter program or data words into consecutive memory cells starting from a specified address XXXX

E XXXXC	XXXX XXXX XXXX XXXX etc	clear screen first, then enter begin
E XXXX	XXXX XXXX XXXX XXXX etc	same as above line w/o clearing screen.

3. INSPECT & CHANGE   allows user to inspect and change a specific memory cell if desired.

```
I XXXX   XXXX XXXX      current contents.
         XXXX XXXX DONE   new contents.
```

4. EXECUTE   loads registers with contents from RAM (0FF0-0FFF) and jump to specified address.

X XXXX

5. PUNCH PAPER TAPE   punches a paper tape of a specified memory area with checksum, leader, and tail.

P XXXX THRU XXXX   DONE

6. READ PAPER TAPE   reads the contents of a paper tape into a memory area specified on tape. Checks checksum on tape and prints checksum error message if one was made.

R XXXX THRU XXXX   DONE

7. CHECK READ PAPER TAPE compares the contents of a paper tape versus the contents of an area in memory and prints out any differences on the CRT. This routine also compares and checks the checksum on the tape and prints out a checksum error if one was made.

C XXXX THRU XXXX GOOD TAPE!

8. LIST ON LINE PRINTER lists (dump) an area of memory to the line printer for a hardcopy listing.

L XXXX THRU XXXX DONE

9. STORE CONSTANT stores a 32-bit constant into a specified area of memory.

S XXXX THRU XXXX W-XXXX XXXX DONE

10. FIND (SEARCH) searches a specific area in memory for a constant MASKED, and prints out all locations where found.

F XXXX THRU XXXX W-XXXX XXXX M-XXXX XXXX

11. MOVE moves one block of memory from one location to another.

M XXXX THRU XXXX NEW LOC XXXX DONE

12. BREAKPOINT BREAKPOINT ROUTINE same as execute routine, but dumps registers and stack and goes to "RDY" if the program reaches the breakpoint address.

B XXXX TO XXXX

B R
XXXX TO XXXX (Breakpoint resume)

B =
XXXX M-XXXX XXXX T-XXXX XXXX DONE

(above line: B =, allows user to restore the original content of the breakpoint address in the case where the program does not reach the breakpoint address.)

13. "Clear Screen" clears screen and clears register and flag area in RAM; RESET STACK POINTERS, GOTO RDY

14. "INT" GO TO SP, RDY resets program stack pointer. Sets CRT/LP flag to CRT, GO TO RDY

15. VERIFY Prints out the CHECKSUM of a specified area of Memory.

OTHER FEATURES

FAULT ROUTINE

Any time a ØØ instruction is fetched, the Program Counter will be forced to location ØØØØ and this fault routine will be executed. This routine prints out (on the terminal) the location of the fault, the contents of the fault location, a dump of the current register at the time of the fault, and a snapshot dump of the memory area where the fault occurred. At the end of this snapshot dump, the Program Control will be at the end of the DUMP MODE so the user may immediately inspect the area where the fault occurred and then use the monitor functions to change the program.

ADDITIONAL EXECUTIVE ROUTINES

The user is enabled to insert additional executive routines in RAM other than those just discussed. To implement this feature, the user must place, in the lower half of the memory location ØFEB, the address where the check for the additional executive routine character is located. If the input character does not match any additional executive routine function character, the Program Control should go to READY.

BACKGROUND ROUTINE

A background program, a program that runs when the MIMC is in a state to receive input from the keyboard, may be executed by placing, in the upper half of memory location ØFEB, the beginning address of the background program. The background program must be perfect in the sense that, before it returns to the keyboard poll routine, all registers except, R1, Q, and SR, must be restored to their original values. The stack and stack pointer must also be the same. The background program returns to location ØØ8E to poll the keyboard for input.

When a background program becomes fully debugged, it may be placed in PROM. The starting address of this background program may also be placed in PROM (lower half of location Ø579). This enables the background program always to be in memory and running without having to load it or its starting address into RAM each time the MIMC is powered up.

A routine by which more than one background routine may be running consecutively, one after another (if no input from the keyboard is received), has been developed and tested. This routine, along with additional background programs, could be added to PROM quite easily. Memory location ØFEB is used as a switch to go from one background program to another.

The beginning background routine address in RAM (location ØFEB) overrides any beginning background routine address stored in PROM. The beginning background routine address in PROM is seen by the monitor only when the upper half of memory location ØFEB is ØØØØ. If the upper half of memory location ØFEB is not zero, the monitor will jump to the location specified by the upper half of location ØFEB, thinking a background program exists in that location.

If the upper half of location ØFEB and the lower half of the location where the beginning background program address is stored in PROM are both zero, no background program will be executed. The monitor will remain in a loop, polling for input from the keyboard.

SECTION 4. THE CCD MODULE

GENERAL CHARACTERISTICS

In the design of the CCD module, many factors (sometimes conflicting factors) were taken into account. The first of these was the need for a highly reliable system since the CCDs will store programs and, if bits of an instruction are erroneously changed, the whole system could crash. Assurance of hardware reliability can best be obtained by employing conservative design, using low power and a minimum number of components, providing regular maintenance (automatic as well as operator driven), and incorporating automatic Error Detection and Correction (EDAC).

Modularity is an important consideration. Modularity allows the memory to be scaled in size so that it is useful to many systems with varying storage requirements. The CCD module is addressed using a 12-bit selection code which provides more than sufficient expandability.

The current emphasis on hardware standardization encourages the use of the Navy Standard Hardware Program (SHP) 40-pin connector and modular card-size increments. The CCD module uses the same Super 2A cards as does the MIMC described in the previous section. The use of this card imposed some restrictions and caused some problems (as would any standard), but useful hardware was designed.

The overall function of the CCD module is to store data in a small volume at a cost which is competitive with a drum or a fixed-head disk (FHD) while providing five to ten times faster access to a block of data than is possible with a drum or disk. With the MIMC or similar controller, the access time to a 256-word block of data is between 600 and 700 microseconds. The volume occupied by the module is 270 cubic inches (4425 cm^3). In production, a 16-million bit memory (four modules) should cost about 35 thousand dollars or two tenths of a cent per bit. This cost includes the controller, power supplies, an NTDS interface, EDAC, and a cabinet suited to the military environment.

The CCD memory module consists of 24 Super 2A cards. Twenty-two of these, the CCD memory cards, are identical. The remaining two cards are the Memory Segment Control (MSC) card and the Data Error Detection and Correction (DEDC) card. The module stores 262 144 words of data. Each word comprises 16 data bits and six EDAC bits. The EDAC bits allow the module circuitry to detect and correct any one-bit error in the 22-bit word, to detect but not correct any two-bit error in any word, and to detect many errors in which an odd number of bits are in error. (A modified Hamming code of distance four is used.)

A RAM buffer has been incorporated into the module so that, once the desired block has been written from CCD into the buffer, any word of the block can be accessed. The use of the RAM buffer also guarantees that the access time to a block of data (256 words) is always the same, about 650 microseconds. This is true because it is not important where the first word of the block is located relative to the output of the 256-bit CCD register. Whatever word is available for readout is written into its appropriate location in RAM immediately after the request for that block is received by the module. Each word is then read out in sequence until all of the 256 words have been read out of the CCD and written into RAM. This takes about 650 microseconds. This fact, which leaves no doubt as to when the data will be available, simplifies system timing.

THE ARCHITECTURE

The architecture of the CCD module was influenced by several factors. One of the most important of these was the desire to keep the CCD module as independent as possible of the controller. It was desired that minimum control be required. Refreshing of data in the CCDs proceeds independently of the controller. Also, word-by-word transfer of data from CCD memory does not require controller action. A RAM buffer is used for temporary block storage and allows a definite access time (the time required to transfer a block of 256 words) to the RAM from the CCD. The EDAC function was placed inside the module so that errors in CCD could be detected and corrected before the rest of the system could be affected. Also, it was determined that the speed required to perform a word-by-word check could best be achieved by hardware.

A major point in the architecture of the module is the use of the bit-slice approach. Each of the CCD memory cards contains one bit of all the 1024 blocks (262 144 words) in the module. This bit-slice approach allows flexibility in the design of the system. For example, for the ultimate in reliability, a word length of 22 bits is used (22 cards) which provides automatic error correction for single-bit errors. If this feature is found to be too expensive, five of the code bits can be eliminated retaining only the parity bit which would give an indication that an odd number of bits had changed. Checksums computed by the controller could be used as a further check on data integrity. Eliminating the error-correction feature would reduce memory system costs by about 20 percent. Further, the system could also be used without parity, requiring only 16 cards. In this case, the controller would be relied upon to detect errors. If error rates are found to be sufficiently low, this approach may be satisfactory to many systems.

Other factors considered in the definition of the module included physical limitations (thermal, mechanical, electronic), availability of CCD components, pin limitations on connectors, the Super 2A cards, modularity of the system, reliability, and other subtle factors derived from the systems experience of several engineers.

The simplest view of the CCD module architecture is shown in figure 4. The system data bus carries 16 bits of data into the DEDC card. The system address bus and control lines feed signals into the Memory Segment Control (MSC) card. These two cards drive the CCD memory bus. Each CCD card is one bit of the 22-bit word.

The MSC card, shown in figure 5, has several sections. The CCD module main control section interfaces with the system address and control lines to determine whether that particular module is being selected. Decoding of function codes and sequencing for block transfers are also controlled in this section. The RAM buffer control section interfaces the system bus, or the CCD memory, to the 256-by-16 bit memory on the DEDC card. The CCD memory clock and control section generates the signals for block transfers to and from the CCD cards.

The CCD memory-refresh clock section must be active at all times in order that data may be maintained. This section is independent of the other sections in that power can be removed from the latter and this section will continue to function, supplying refresh clocks to the CCD cards. This section contains a word counter which maintains the phase of the CCD register, determining which of the 256 words is available at a given time.

The DEDC card, figure 6, contains the RAM buffer, buffer-addressing circuitry, status registers and EDAC circuitry, and has drivers and receivers. Data are transmitted to or from the system through tri-state transceivers. Incoming data are written into RAM or used as RAM or block addresses. RAM addresses are routed from the system bus, on to the RAM bus, through the RAM address multiplexer, to the RAM address counter/latch. Block

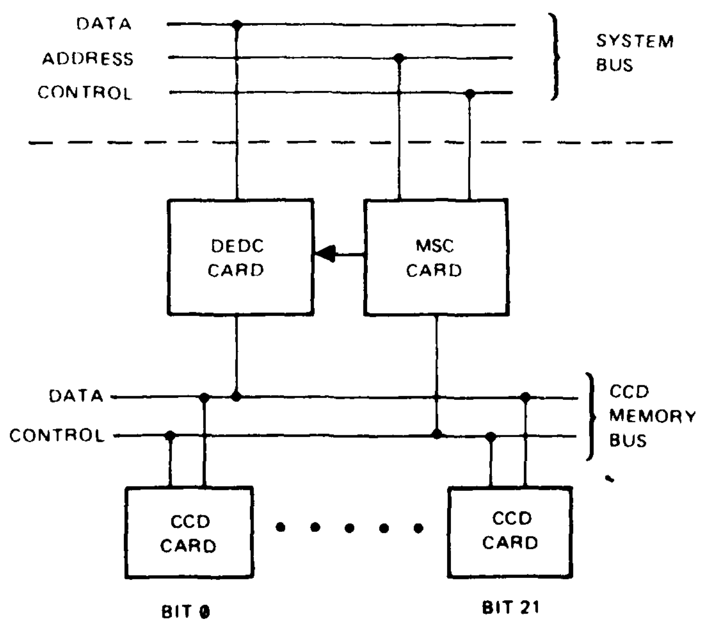


Figure 4. CCD module architecture.

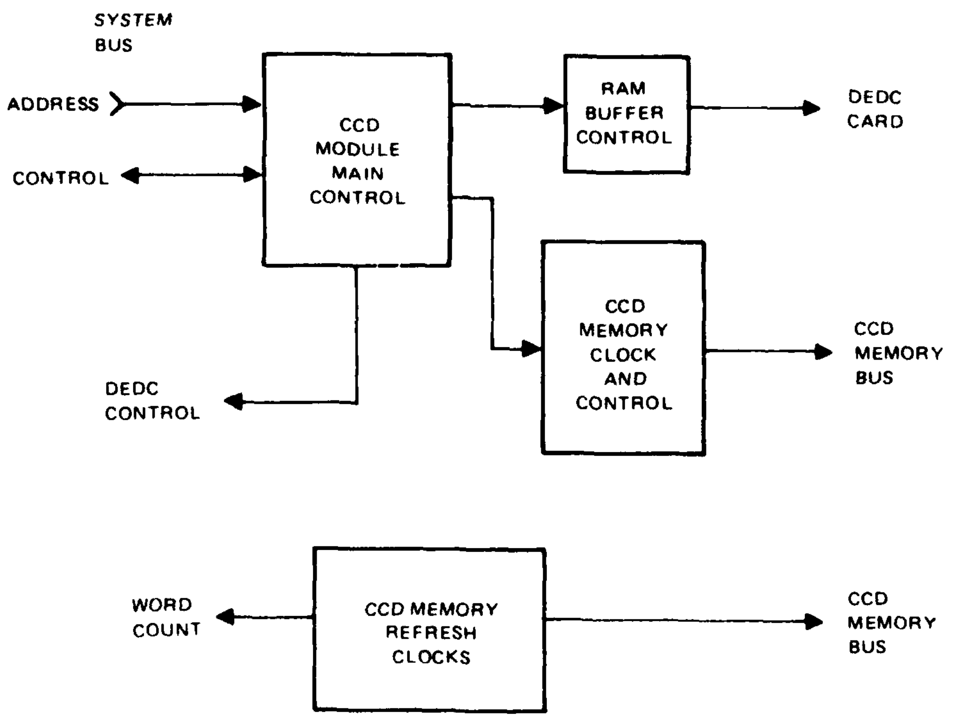


Figure 5. Memory Segment Control (MSC) card block diagram.

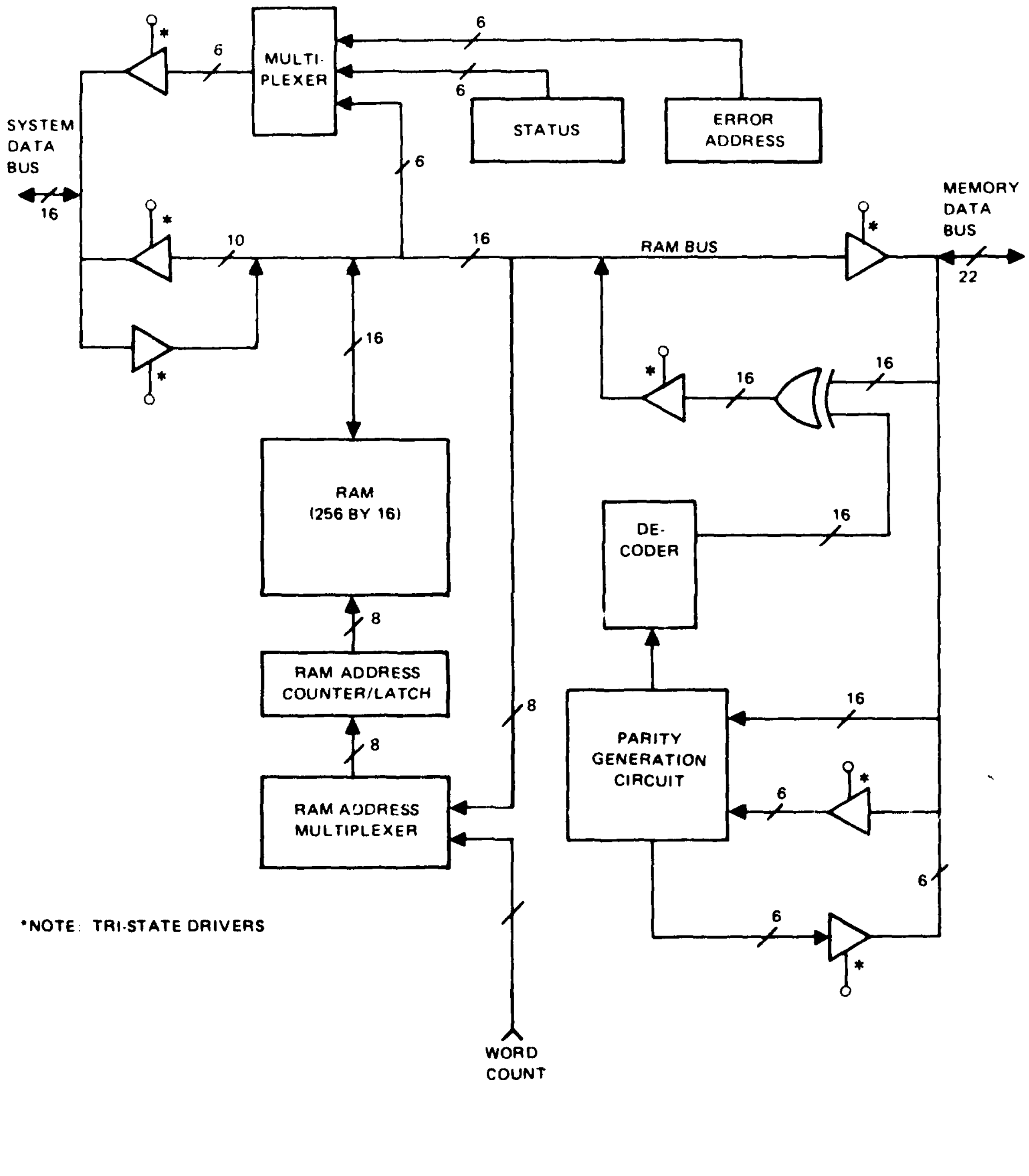


Figure 6. Data/Error Detection and Correction (DEDC) card block diagram.

addresses are routed from the system bus, on to the RAM bus, through the CCD memory bus drivers, on to the CCD cards where they are latched into the block address registers.

When data are read out of the RAM on to the CCD memory bus, the 16 bits are also fed into a parity-generation circuit which generates six code bits. These six code bits, together with the 16 data bits, make up the 22-bit memory bus. More detail on this operation will be presented when FDAC is discussed.

When data and code bits are read from the CCD cards, the 16 data bits pass through exclusive OR gates, through tri-state bus drivers, on to the RAM bus, and into the buffer. If the code bits indicate an error in one of the data bits, the parity generator/checker network will send an address to the decoder which will cause one of the data bits to be inverted in passing through the exclusive OR gate on its way to the RAM.

The CCD memory card block diagram is shown in figure 7. This card is made relatively simple in order to keep costs from rising. This is the most populous board in the system and, for this reason, has a strong influence on system cost. There are 16 CCD devices (Intel-2416) on the card. The inputs of all of these devices are tied together as are their outputs. When the block address (10 bits) is latched into the block-address latch, six of these lines go to all of the CCDs and four go to the decoder. In this manner, only one block is selected (bits zero to 5) and only one chip is selected (bits 6 to 9).

Note that four chips run off one of the four multiplexers. This permits keeping the temperature of the card low since only one-quarter of the chips on any one card will be running at high speed during reading or writing. Since the ratio between standby clock and transfer clock could be three or four to one, this multiplexing permits a significant power saving. This is a particularly important factor in this system where a dynamic memory is used and which is affected adversely by temperature in its data-retention function (as well as reliability of the device itself). During a power outage, all unnecessary circuits are turned off to minimize drain on batteries used to maintain the data.

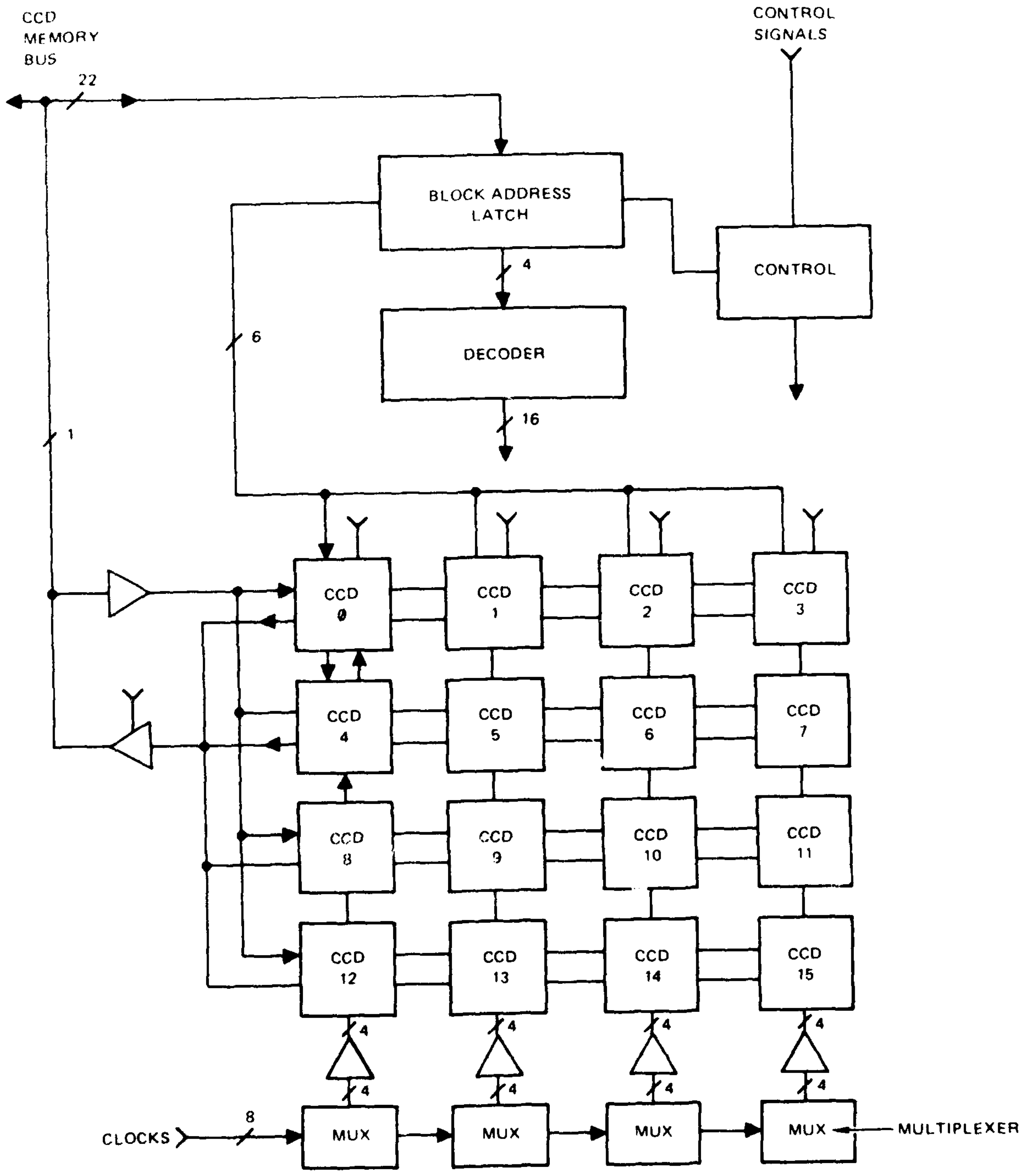


Figure 7. CCD memory card block diagram.

FUNCTION CODES

The CCD module is controlled by the address bus, the system data bus, and control lines. The present hardware lines use six bits of selection code to select one of 63 modules. (The all-zero address should be reserved for other input-output addresses.) Four bits of the 16 address lines in the system are used for commands. Table 3 shows the various commands which can be given to the module.

TABLE 3. MEMORY SEGMENT FUNCTION CODES.

Code_{16}	Function
Ø	Write word into address A* of RAM buffer.
1	Read word from address A* of RAM buffer.
2	Write block B* into RAM buffer from CCD.
3	Read block B* from RAM buffer into CCD.
4	Undefined.
5	Undefined.
6	Undefined.
7	Undefined.
8	Read status.
9	Read error syndrome.
A	Enable interrupt request.
B	Disable interrupt request.
C	Undefined.
D	EDAC disable.
E	Undefined.
F	Master clear memory segment.

*The RAM address A or the block number B is placed on the system data bus during the I/O instruction.

CODE $Ø_{16}$ SINGLE-WORD WRITE INTO RAM BUFFER

The particular module is selected (bits 4 to 9) and the function code of Ø (bits zero to 3) is specified by placing the appropriate value on the address lines. The RAM buffer address (bits zero to 7) is placed on the system data bus and the write line is pulsed. The RAM address will be loaded into the RAM address latch with the falling edge of the bus time signal. No bus wait is generated by the MSC. The data to be written into the RAM address are placed on the system data bus (bits zero to 15) and the write line is pulsed. Bus wait is active long enough to complete the write cycle.

CODE 1_{16} SINGLE-WORD READ FROM RAM BUFFER

The particular module is selected (bits 4 to 9) and the function code of 1 (bits zero to 3) is specified by placing the appropriate value on the address lines. The RAM buffer address (bits zero to 7) is placed on the system data bus and the write line is pulsed. The

RAM address is loaded into the RAM address latch with the falling edge of the bus time signal. No bus wait is generated by the MSC. The data are read from the system data bus (bits zero to 15) by pulsing the read line. Bus wait is active long enough to guarantee valid data from the RAM buffer.

CODE 2_{16} WRITE BLOCK B INTO RAM BUFFER FROM CCD

The particular module is selected (bits 4 to 9) and the function code of 2_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. The block number B is placed on the system data bus (bits zero to 15) and the write line is pulsed. The block number is loaded into the latches on the CCD cards by the falling edge of the bus write signal. No further action is required by the controller. The MSC immediately raises the active line and performs the necessary operations to complete the transfer of the requested block into the RAM buffer. During the transfer, the transfer line is high. The active line goes low when the CCD registers have been resynchronized.

CODE 3_{16} READ BLOCK B FROM RAM BUFFER INTO CCD

The particular module is selected (bits 4 to 9) and the function code of 3_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. The block number B is placed on the system data bus (bits zero to 15) and the write line is pulsed. The block number is loaded into the latches on the CCD cards by the falling edge of the bus write signal. No further action is required of the controller. The MSC immediately raises the active line and performs the necessary operations to complete the transfer of the requested block into the CCD. During the transfer, the transfer line is high. The active line goes low when the CCD registers have been resynchronized.

CODE 4_{16} (UNDEFINED)

CODE 5_{16} (UNDEFINED)

CODE 6_{16} (UNDEFINED)

CODE 7_{16} (UNDEFINED)

CODE 8_{16} READ STATUS

The particular module is selected (bits 4 to 9) and the function code of 8_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. The status bits are placed on the system data bus (bits zero to 5) when the read signal is active. No bus wait is generated. The six status bits are interpreted as:

NAME	BIT	DEFINITION (ACTIVE HIGH)
Transfer	0	Data are being transferred between the RAM buffer and CCD
Active	1	Data are being transferred or the CCD registers are being resynchronized (do not initiate another block transfer).
Error	2	At least one error was made during the last block read
Corrected	3	At least one error was corrected during the last block transfer.
Detected	4	At least one word had a two-bit error during the last block read.
Ready	5	A CCD refresh cycle has been completed.

CODE 9_{16} READ ERROR ADDRESS

The particular module is selected (bits 4 to 9) and the function code of 9_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. The error-address bits (zero to 5) are placed on the system data bus when the read signal is active. No bus wait is generated. The error address should be interpreted as follows (hexadecimal):

ERROR ADDRESS (HEX)	BAD DATA BIT POSITION (DECIMAL)
Ø3	Ø
Ø5	1
Ø6	2
Ø7	3
Ø9	4
ØA	5
ØB	6
ØC	7
ØD	8
ØE	9
ØF	10
11	11
12	12
13	13
14	14
15	15

The error addresses Ø1, Ø2, Ø4, Ø8, and 1Ø refer to code bits in error. Code bits are not corrected but are regenerated when data are read out of the RAM buffer into the CCD. Error address Ø means no errors. Error addresses 16 through 3F have no meaning in this system other than that their appearance may indicate a possible hardware failure.

CODE A_{16} ENABLE INTERRUPT REQUEST

The particular module is selected (bits 4 to 9) and the function code of A_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. No data are required on the system bus. When the write signal is pulsed, the flip-flop which allows the MSC to interrupt the controller is set to allow interrupts.

CODE B_{16} DISABLE INTERRUPT REQUEST

The particular module is selected (bits 4 to 9) and the function code B_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. No other data are required on the system data bus. When the write signal is pulsed, the flip-flop which allows the MSC to interrupt the controller is cleared to prohibit interrupts.

CODE C_{16} (UNDEFINED)

CODE F_{16} MASTER CLEAR MEMORY SEGMENT

The particular module is selected (bits 4 to 9) and the function code F_{16} (bits zero to 3) is specified by placing the appropriate value on the address lines. No data are required on the system data bus. When the write signal is active, the following occurs.

Interrupts are disabled;

EDAC is enabled;

The RAM address counter/latch is cleared;

The error address latch is cleared;

The error status (error made, error corrected, errors detected) latch is cleared;

The block number registers on the CCD memory cards are cleared; and

Other miscellaneous flip-flops are also cleared to initialize the module.

EDAC OPERATION

In this brief description of the operation of the EDAC circuitry, no attempt will be made to go into the theory of coding. The reader is referred to the many books on the subject for further information.

In simple terms, in this system, even parity is computed over various groups of the 16 data bits. Table 4 shows the relations between the code bits, the data bits, and memory-bus bit position (zero through 21). Referring to the table, note that code bit Ø is the bit which makes the parity of the bits in that group even. This means, if data bits Ø, 1, 3, 4, 6, 8, 1Ø, 11, 13, and 15 are all ones, code bit Ø must be a zero to maintain even parity in that group. In other words, ten ones is an even number of ones and, for that group to maintain even parity, the code bit must be a zero. If one of the data bits is a zero, leaving nine ones, the code bit would have to be a one to obtain even parity over the group CØ, DØ, D1, D3, D4, D6, D8, D1Ø, D11, D13, and D15. Similarly, with the other groups, 1, 2, 3, and 4, the code bit is computed so that the group which includes the code bit has an even number of ones.

The code bit C5, however, is computed so that parity over the entire 22 bits is odd: an odd number of ones. For instance, if all the data bits were zero, code bits CØ, C1, C2, C3, and C4 would also be zeroes. To achieve odd parity over the entire word, C5 must be a one. In this case there will be an odd number of ones, namely one, in the 22 bits.

TABLE 4 MEMORY BUS DATA CODE PARITY BIT RELATIONS

MEMORY BUS NUMBER	16	17	0	18	1	2	3	19	4	5	6	7	8	9	10	20	11	12	13	14	15	21
TOTAL POSITION NO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
CODE BIT	C0	C1		C2				C3								C4						C5
DATA BIT			D0		D1	D2	D3		D4	D5	D6	D7	D8	D9	D10		D11	D12	D13	D14	D15	
GROUP 0	Ⓧ		X		X		X		X		X		X		X		X		X		X	
GROUP 1		Ⓧ	X			X	X			X	X			X	X			X	X			
GROUP 2				Ⓧ	X	X	X					X	X	X	X					X	X	
GROUP 3								Ⓧ	X	X	X	X	X	X	X							
GROUP 4																Ⓧ	X	X	X	X	X	
TOTAL WORD	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Ⓧ

Note Parity is "even"* over each group 0, 1, 2, 3, 4.

Parity is odd over the entire memory word.

Error address is C4, C3, C2, C1, C0. It is obtained by exclusive ORing the code generated from the returning data and the error code stored with the data.

*An even number of "ones" means P = 1 (0, 2, 4, etc, 1s).

The outputs from the parity generator checkers are fed to bus-driver circuits and the code bits are stored in CCD memory along with the data bits. During the reading of a block from CCD memory, the data bits are again fed into the parity generator checker circuits but now the code bit from CCD memory is also fed into the parity circuit. If there was no error, the parity generator checkers would output zeroes to the error address latch.

If an error occurred, the bit position would be indicated by the error address register. This also can be seen in table 4. Note that the parity groups have been selected in "binary" fashion. For example, group 0 is every other position, group 1 uses every other pair of bits, and so forth. Hence, each code bit is a binary address bit. For this reason, when the stored and received bits do not agree, the parity generator checker outputs designate to what groups that bit belonged, resulting in a binary address. Further analysis requires a detailed study of the circuitry which is beyond the scope of this report.

Only the first error correction or detection in a block transfer is indicated in the status bits or in the error address register. (The EDAC bits are cleared by using the Master Clear Memory segment function code F_{16}.) Subsequent corrections will be performed but the status registers will not change until either they are cleared by the Master Clear or by the next block transfer.

SECTION 5. TESTING THE CCD MEMORY MODULE

THE TEST SYSTEM

Because of funding limitations, only a partially populated module was assembled and tested. The MSC and DEDC cards were assembled just as they would be for a full module. The CCD card was modified so that data were entered in parallel into the 16 CCDs, two cards give a full 22 bits, 64 blocks deep. In effect, one-sixteenth of a module was available for test.

This module was imbedded in the MIMC chassis. The entire card cage is shown in figure 8. The MSC, DEDC, and parallel CCD cards are shown in figure 9. These cards are special, high-density, Super 2A size, weld-post cards. In a production system, they would be multilayer printed-circuit boards. Other cards are shown in figures 10, 11, and 12. This chassis, power supplies, a paper-tape punch/reader, and a cathode ray tube made up the test system.

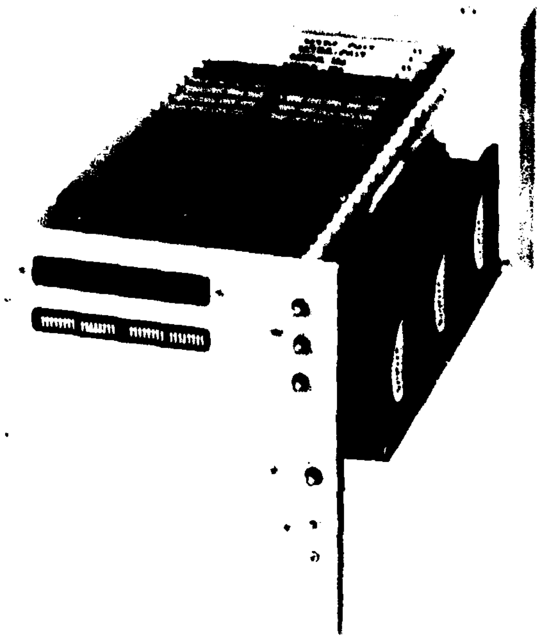


Figure 8. The MIMC chassis card cage.

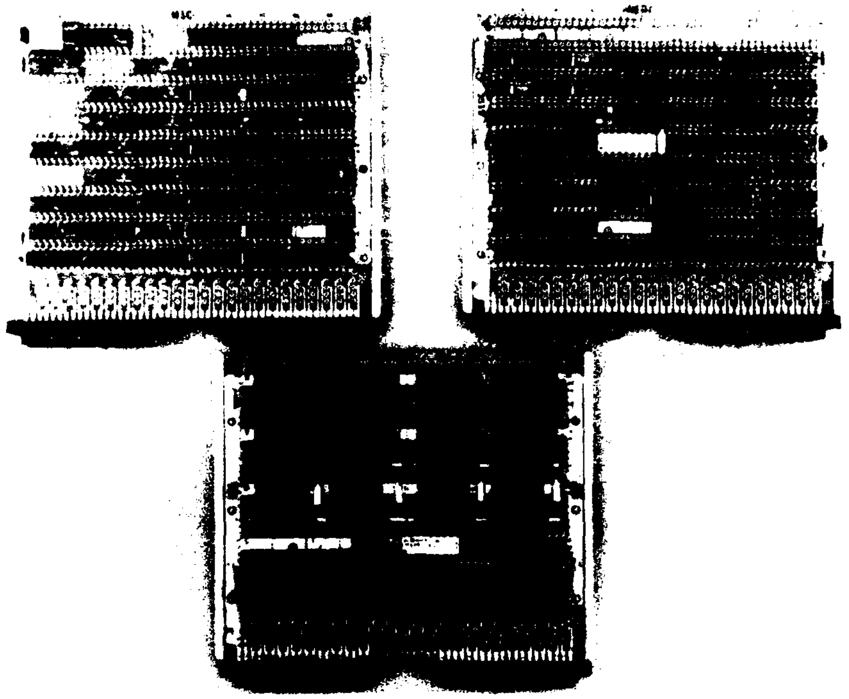


Figure 9. The MSC, DEDC, and CCD cards.

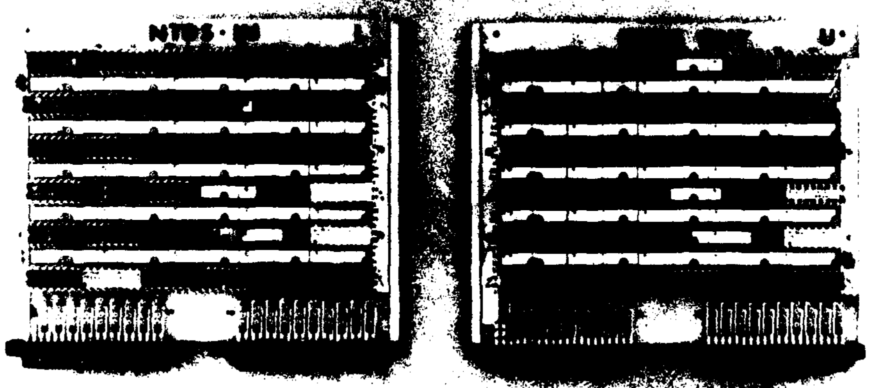


Figure 10. The NTDS input and output cards.

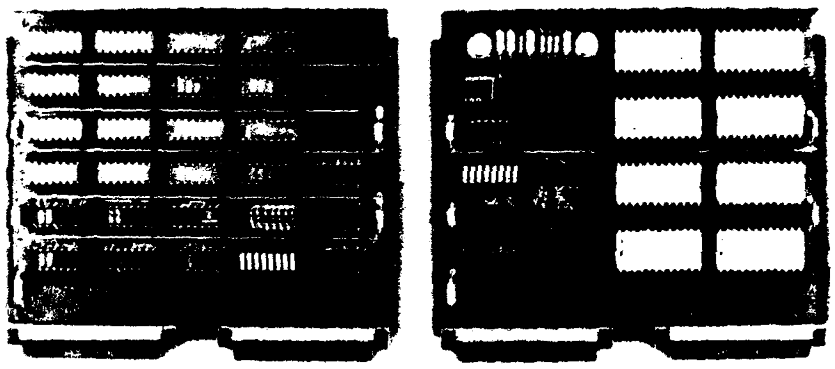


Figure 11. The 2k-byte RAM and 2k-byte PROM cards.

Figure 12. The basic input/output and the REMEX paper-tape reader/punch cards.

TYPES OF TESTS

Early development work on CCD concerned itself with the charge-transfer efficiency and the bit-error rate which could be achieved with existing technology. Early devices which were tested showed serious pattern and temperature sensitivities. For this reason, in testing the developed system, as many patterns as possible were run through the system to determine sensitivities. The patterns chosen for testing were:

Test Ø, all zeroes;

Test 1, all ones,

Test 2, count and inverse;

Test 3, alternating Øs and 1 (checkerboard);

Test 4, galloping 1s (ones);

Test 5, galloping Øs;

Test 6, pseudo-random number and inverse; and

Test 7, half-zero/half ones and inverse.

TESTS Ø AND 1

The all ones and all zeroes tests are simple starting points. No particular problems nor insights were expected or obtained.

TESTS 2 AND 3

The count pattern runs from Ø to 255 in the lower eight bits and from 255 to Ø in the upper half word. This pattern exercises each bit to varying degrees from alternating ones and zeroes to a string of zeroes and a string of ones.

TESTS 4 AND 5

The alternating ones and zeroes present a kind of worst case where every bit is different from its neighbor. The galloping-one and galloping-zero pattern has a single bit in a sea of its complement. This test is expected to show up weaknesses in sense amplifiers and in leakage problems.

TEST 6

The pseudo-random test uses a maximal length (no repeats) 16-bit number to determine sensitivity to a random pattern.

TEST 7

This test stores 128 ones and then 128 zeroes into the 256-bit CCD register. This test was suggested by Dennis Amonson of UNIVAC after he found some sensitivity to that pattern. (This test is the same as the pattern seen by bit 7 of a 256-word count.)

THE TEST PROGRAM

The test program has the following features:

The starting date and time can be stored so long periods can be run with an accurate record of the error rates;

Any combination of the eight tests can be utilized;

Information can be stored and displayed on the request of the following

start date and time,

tests being run,

total number of passes,

total number of errors (bad bits),

the number of uncorrected one-bit errors in a word,

the number of errors which occurred in a greater than two-bits-per-word manner,

the number of blocks in which at least one bit was corrected,

the number of blocks in which two-bit errors were detected,

the number of times an illegal address was read,

the number of errors per test for each of the eight tests,

the number of errors in each bit of the 22 bits, and

the number of errors in each of the 64 blocks;

The first 128 words with errors are stored along with the block number and RAM buffer address;

The statistics can be printed out on the cathode-ray tube at any time the "S" key is depressed; and

The test will continue when the "C" key is depressed.

TEST RESULTS

The results of the testing can be summarized as follows. During 1207 hours of continuous running, from 13 May through 8 July 1976, only ten errors were detected and corrected by the module. At the reading rate used (about 6.8×10^8 bits per hour), this indicates an error rate for the CCDs of 1.2×10^{-11} error per bit read. The system error rate is even better; all of these errors were single-bit errors and were corrected before the data were read into the RAM buffer.

It was particularly gratifying, when the tests were first started, to find a "weak" device. This was possible since the tests kept indicating that a particular bit was bad in a particular block. Replacement of the chip eliminated the error.

It should be noted that all of the tests were conducted at approximately 25°C (room temperature). Problems in other hardware precluded testing at higher temperatures. Such testing should be done.

No pattern sensitivity was observed. The sensitivity observed by UNIVAC personnel was on early devices. The problem was known by Intel and was probably eliminated by modification of their chip-fabrication process.

While the CCD module appears to work very well, there have been some intermittent noise problems with the controller. Improvements in card layout and in bus signals should eliminate these problems.

SECTION 6. SUMMARY AND RECOMMENDATIONS

A modular CCD memory subsystem, useful for Navy C^3 applications, with an intelligent front end has been designed and tested. Basic software has been developed and a monitor has been written to simplify programming. The CCD module has been designed and a partially populated module has been tested. Initial tests showed an error rate for the CCDs of about 1×10^{-11} per bit. The module itself corrected all of the errors. The system is ready for use as a disk or drum replacement. Engineering will be required to expand the bus system and to configure the hardware to a particular cabinet Further testing is required to determine the effects of temperature on the system. UNIVAC has done some temperature cycling and has found adequate performance over the range specified for the Intel device.

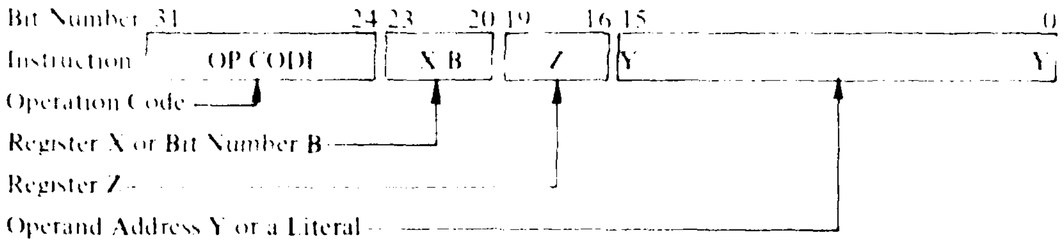
The question of volatility arises with a CCD system. Two answers are available: dump the data on to tape under battery power; and use a battery backup for the CCDs. The first solution answer is somewhat less desirable since it means adding another mechanical device to the system. The second solution answer is feasible but seems to be poorly accepted by systems designers. It may require several years for batteries to become acceptable.

A concept which may minimize the number of batteries takes advantage of the fact that, when the main power is lost in a system, heat generation from electronics operation stops. Thus, when this happens the temperature will drop to approximately ambient (25° to 30°C). Since the CCD specifications require a 50-kHz clock to maintain data at 70°C, the clock required at 30°C may be as low as 12 kHz or one quarter of the normal operating clock rate. For this reason, the number of batteries could be reduced and temperature-sensing circuitry could be incorporated to ensure the correct clock rate.

Rough calculations indicate that, using the slow clock, the data in the module (5.6×10^6 bits) could be maintained for two hours with six D-size, sealed, lead-acid cells. The batteries would require a volume increase of only about 12 percent. This breaks down to 302 cubic inches (4719 cm^3) for the CCDs, and to 30 cubic inches (469 cm^3) for the batteries. Investigation should be continued along these lines.

APPENDIX A: DESCRIPTION OF MIMC INSTRUCTION FORMAT AND DEFINITIONS

The instruction format is:



The instruction mnemonic will be a four-letter symbol for the instruction. The first two letters usually designate the class of instruction, such as, LD for load, AN for AND, and so forth. The third letter signifies that an X register is used in the instruction. The fourth letter designates the type of addressing used to obtain the operand, as listed below. There are exceptions.

Symbol	Description
Q	The Q or extended working register
Z	A register - Ø - F
L	Literal - Y
N	Normal - (Y)
J	Indirect - ((Y))
M	Indexed Literal - Y + (Z)
I	Indexed Normal - (Y + (Z))
K	Indexed Indirect - ((Y + (Z)))

REGISTER DESIGNATION

NO	HEX	LABEL	DESCRIPTION
1	Ø	WR	Working Register
2	1	WR	Working Register
3	2	WR	Working Register
4	3	WR	Working Register
5	4	WR	Working Register
6	5	WR	Working Register
7	6	WR	Working Register
8	7	WR	Working Register
9	8	WR	Working Register
10	9	WR	Working Register
11	A	WR	Working Register
12	B	WR	Working Register
13	C	TR	Test Register
14	D	SP	Stack Pointer
15	E	OR	Operand Register

REGISTER DESIGNATION (Continued)

NO	HEX	LABEL	DESCRIPTION
16	F	PC	Program Counter
17		Q	Accumulator
18		SR	Status Register

RALU REGISTER SELECTION

NO	MICRO CODE	SOURCE	DEST	SOURCE	DEST
0	ØØØ	X	A	Z	B
1	ØØ1	M	A	Z	B
2	Ø1Ø	X	A	X	B
3	Ø11	M	A	X	B
4	1ØØ	X	A	M	B
5	1Ø1	M	A	M	B
6	11Ø	X	A		B
7	111	M	A		B

STATUS REGISTER DEFINITION

BIT	DESCRIPTION	CODE	HEX
Ø	Overflow	ØØØØ	0
1	Accumulator Full	ØØØ1	1
2	Accumulator Zero	ØØ1Ø	2
3	Spare	ØØ11	3
4	Sign Lower	Ø1ØØ	4
5	Sign Upper	Ø1Ø1	5
6	Carry Lower	Ø11Ø	6
7	Carry Upper	Ø111	7
8	Undefined	1ØØØ	8
9	Undefined	1ØØ1	9
10	Undefined	1Ø1Ø	A
11	Undefined	1Ø11	B
12	Undefined	11ØØ	C
13	Undefined	11Ø1	D
14	Undefined	111Ø	E
15	Undefined	1111	F

The symbols are defined as follows:

SYMBOL	DEFINITION
Q	The Q register.
X	The register designated in the X field (bits 2Ø-23).
Z	The register designated in the Z field (bits 16-19).
Y	The number contained in the lower 16 bits (Ø-15).

SYMBOL	DEFINITION
(Q)	The number contained in the Q register.
(X)	The number contained in the X register.
(Z)	The number contained in the Z register.
(Y)	The number contained at memory location Y.
Y + (Z)	The number obtained by adding Y to (Z).
PC	The Program Counter register.
(PC)	The number contained in PC.
SR	The Status Register.
(SR)	The number contained in SR.
B	The number in the B field (bits 2Ø-23).
SP	The Stack Pointer Register.
(SP)	The number contained in SP.

APPENDIX B. MIMC PRELIMINARY INSTRUCTION SET

CLASS	MNEM	OPCODE	SRE	FUNCTION
LOAD	LDXQ	12		(Q) → X
	LDXL	20		Y → X
	LDXN	21		(Y) → X
	LDXM	22		Y + (Z) → X
	LDXI	23		(Y + (Z)) → X
STORE	STXQ	13		(X) → Q
	STXZ	3Ø		(X) → Z
	STXN	31		(X) → Y
	STXM	32		(X) → Y + (Z)
LOGIC	ANXZ	4Ø	*	(X) Λ (Z) → X
	ORXZ	48	*	(X) V (Z) → X
	XRXZ	5Ø	*	(X) Λ (Z) → X
	CLRX	8Ø		0 → X
	CLMI	83		0 → Y + (Z)
	INVX	1F	*	X → X
ARITHMETIC	ADXZ	AØ	*	(X) + (Z) → X
	ADXL	A1	*	(X) + Y → X
	ADXN	A2	*	(X) + (Y) → X
	SBXZ	A8	*	(X) – (Z) → X
	SBXL	A9	*	(X) – Y → X
	SBXN	AA	*	(X) – (Y) → X
	INCX	9Ø	*	(X) + 1 → X
	DECX	98	*	(X) – 1 → X
SHIFT	SRCX	6Ø	*	(X) is right shifted Y times (circular)
	SRLX	61	*	(X) is right shifted Y times (zero filled)
	SLCX	64	*	(X) is left shifted Y times (circular)
	SLLX	65	*	(X) is left shifted Y times (zero filled)
JUMPS	JUNL	7Ø		Y → PC
	JUNN	71		(Y) → PC
	JUNX	72		(X) → PC
	JNXL	74	*	If (X) = 0, do NI. (X) ≠ 0, Y → PC
	JLXI	76	*	If (X) = 0, do NI. (X) ≠ 0 Y + (Z) → PC, decrement (X)
	JBSL	78		If bit B of SR = 1, Y → PC. If = 0, do NI
	JBSN	79		If bit B of SR = 1, (Y) → PC. If = 0, do NI
	JBZL	7C		If bit B of Z = 1, Y → PC. If = 0, do NI
	JBZN	7D		If bit B of Z = 1, (Y) → PC. If = 0, do NI
COMPARES	CPXZ	CØ	*	(X) – (Z) → Q
	CPXL	C1	*	(X) – Y → Q
	CPXN	C2	*	(X) – (Y) → Q
	MSXZ	C8	*	If (X) = (Z), 0 → X, Skip NI. If (X) ≠ (Z), increment (X), do NI
	MSXL	C9	*	If (X) = Y, 0 → X, Skip NI. If (X) ≠ Y, increment (X), do NI

CLASS	MNEM	OPCODE	SRL	FUNCTION
COMPARES	MSXN	CA	*	If (X) = (Y), 0 → X, Skip NI. If (X) ≠ (Y), increment (X), do NI
INTERRUPTS	INTY	BØ		Interrupt entry: (PC) – 1 → (SP), Y → PC, (SP) – 1 → SP
	ITEX	B1		Interrupt exit: ((SP) + 1) → PC, (SP) + 1 → SP
	ITXR	B2		Interrupt exit and enable interrupts
	CALL	B3		Call subroutine, PC → (SP), (SP) – 1 → SP, Y → PC
	RETN	B4		Return from subroutine, ((SP) + 1) → PC, (SP) + 1 → SP
	RTNR	B5		Return from subroutine and enable interrupts
	DINT	1D		Disable all interrupts
	EINT	1E		Enable all interrupts
STACKING	PSHX	B8		(X) → (SP), (SP) – 1 → SP
	POPX	BC		((SP) + 1) → X, (SP) + 1 → SP
	PSHS	B6		(SR) → (SP), (SP) – 1 → SP
	POPS	B7	*	((SP) + 1) → SR, (SP) + 1 → SP
I/O	OTXY	1Ø		Output (X) to Channel Y
	INXY	11		Input to X from Channel Y
	ISXY	17		Input (slow) to X from Channel Y
MISC	FALT	ØØ		Fault
	INRX	Ø1		Manual inspection of (X) on panel
	ICML	Ø2		Manual inspection of (Y) on panel
	DSPX	Ø3	*	(X) is displayed on the panel for Y/8 seconds. Z is used for counting.
	BSTS	Ø4		Cycles bus control lines. Used for debug only.
	DELY	Ø5	*	Delays next instruction for Y microseconds.

NOTE: The → means that a number is moved to a location. For instance:

(Q) → X means the number in register Q is loaded into register X.

Y + (Z) → X means the number Y plus the number in register Z is loaded into register X.

(Y + (Z)) → X means that the number contained at memory location Y + (Z) is loaded into register X.

((SP) + 1) → X means that the number contained at the memory location (SP) + 1 is loaded into the register X.

(X) → SP means that the number in register X is stored at the memory location designated by the number in SP.

The * means that the Status Register is loaded.

NELC PHOTOGRAPHS

Illustrations in this report include the following NELC photographs.

Illustration No	NELC Photograph No
8	LSF 1163-7-76
9	LSF 1165-7-76
10	LSF 1164-7-76
11	LSF 1167-7-76
12	LSF 1166-7-76